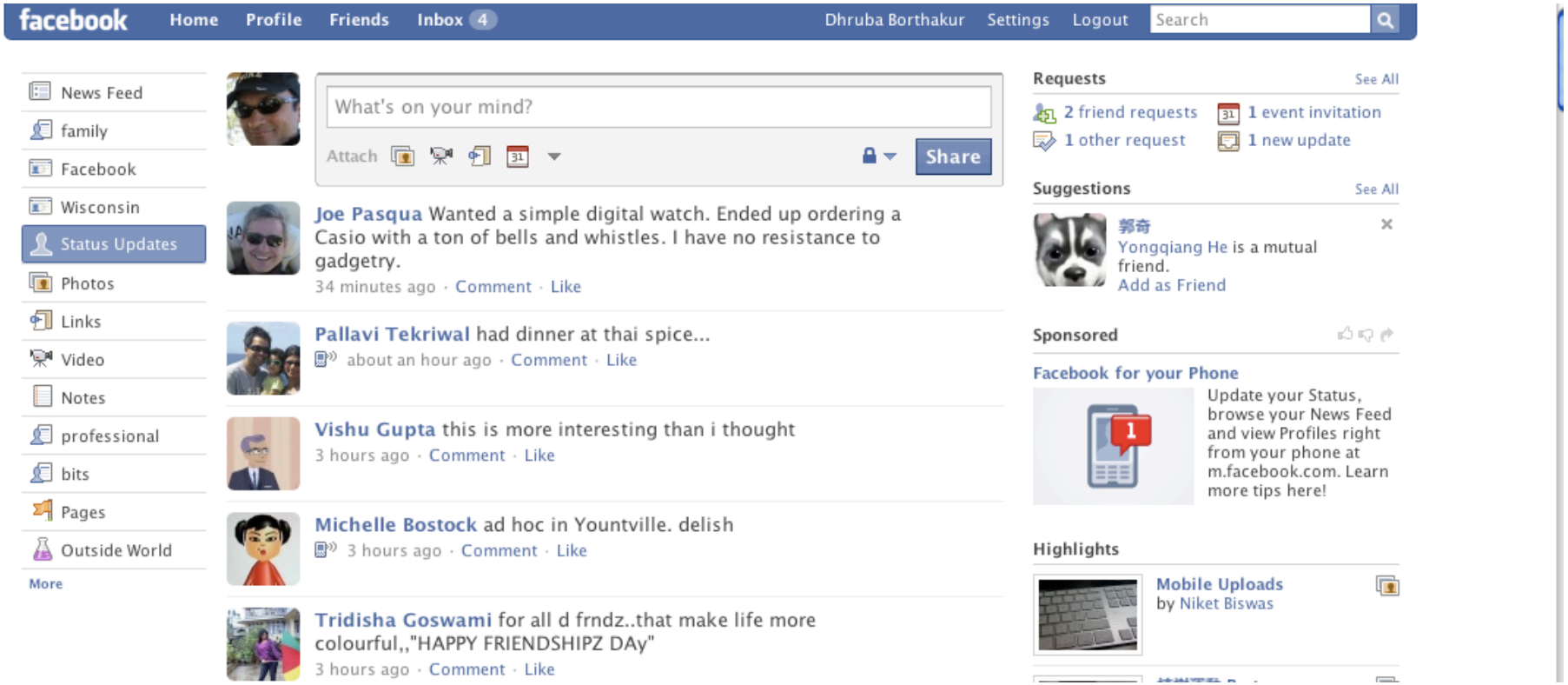
facebook

Hadoop and Hive Development at Facebook

Dhruba Borthakur Zheng Shao
{dhruba, zshao}@facebook.com
Presented at Hadoop World, New York
October 2, 2009

Hadoop @ Facebook



facebook

Who generates this data?

- **Lots of data is generated on Facebook**
  - 300+ million active users
  - 30 million users update their statuses at least once each day
  - More than 1 billion photos uploaded each month
  - More than 10 million videos uploaded each month
  - More than 1 billion pieces of content (web links, news stories, blog posts, notes, photos, etc.) shared each week

Data Usage

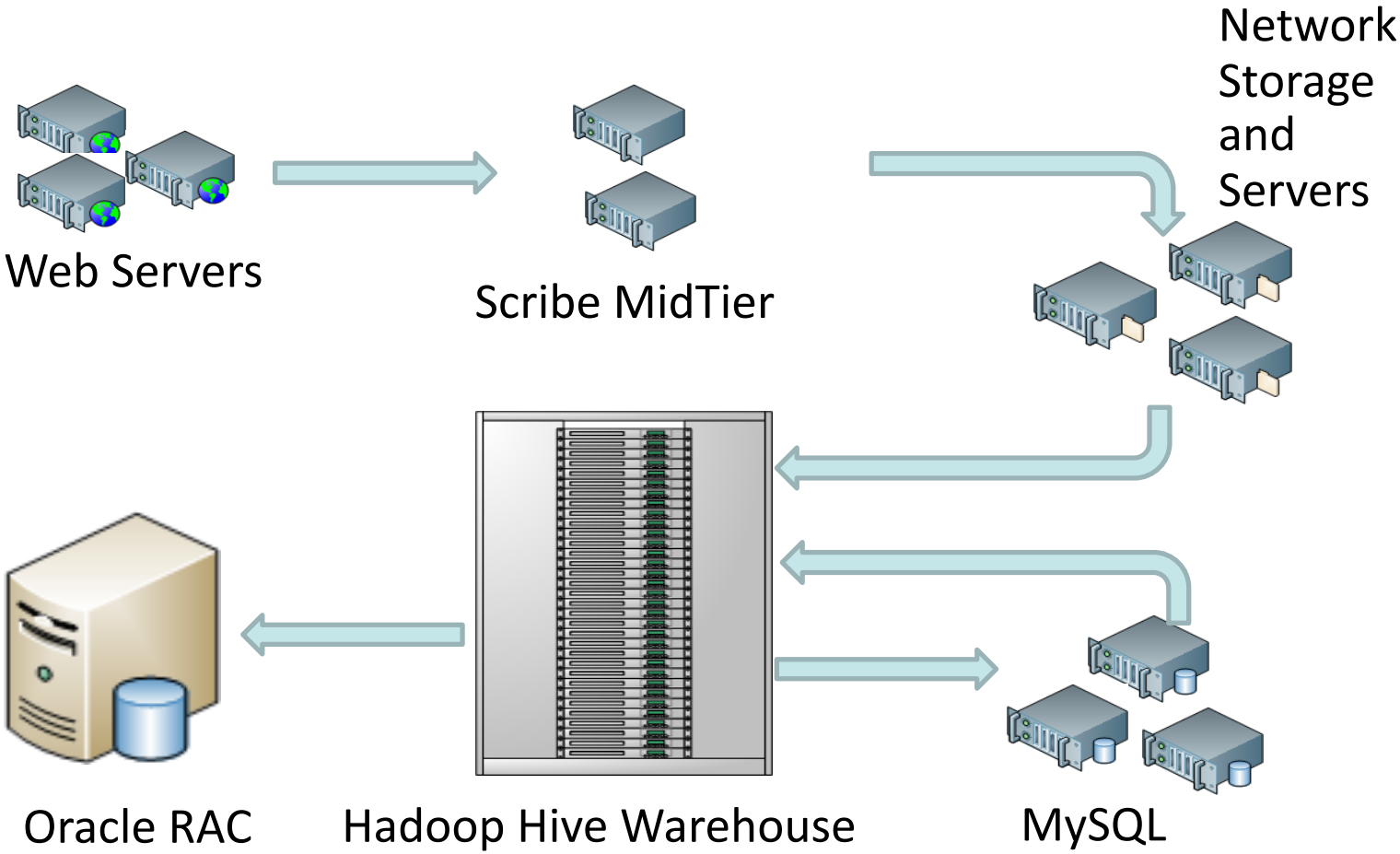
- **Statistics per day:**
  - 4 TB of compressed new data added per day
  - 135TB of compressed data scanned per day
  - 7500+ Hive jobs on production cluster per day
  - 80K compute hours per day
- **Barrier to entry is significantly reduced:**
  - New engineers go though a Hive training session
  - ~200 people/month run jobs on Hadoop/Hive
  - Analysts (non-engineers) use Hadoop through Hive

Where is this data stored?

- **Hadoop/Hive Warehouse**
  - 4800 cores, 5.5 PetaBytes
  - 12 TB per node
  - Two level network topology
    - 1 Gbit/sec from node to rack switch
    - 4 Gbit/sec to top level rack switch

facebook

Data Flow into Hadoop Cloud

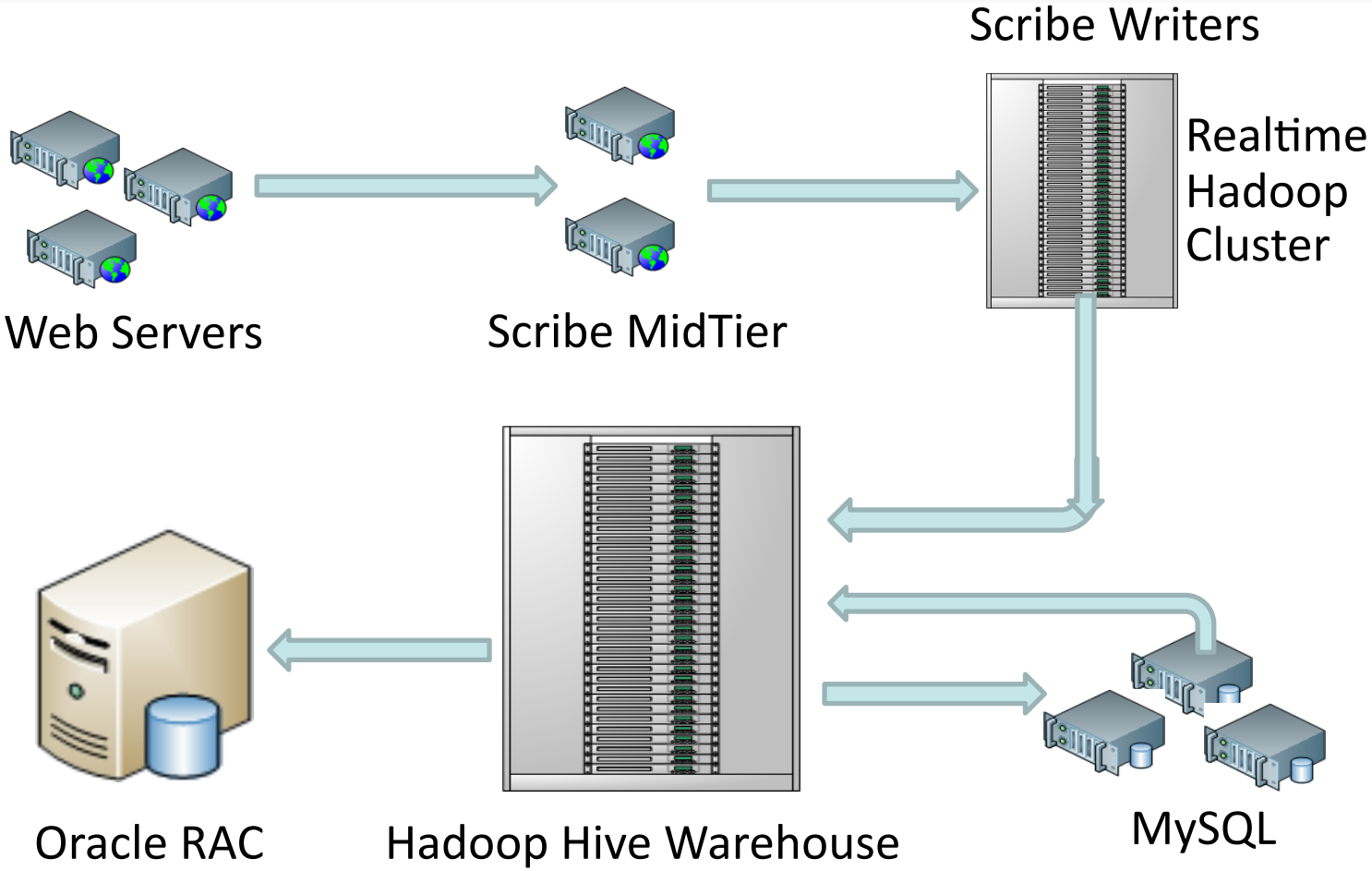


HIVE

hadoop

facebook

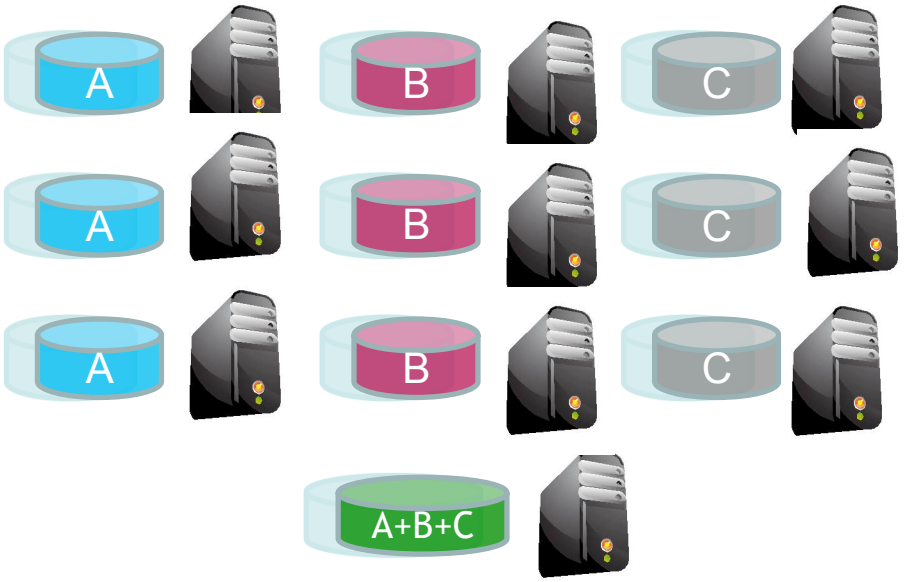
Hadoop Scribe: Avoid Costly Filers



http://hadoopblog.blogspot.com/2009/06/hdfs-scribe-integration.html

HDFS Raid

- Start the same: triplicate every data block
- Background encoding
  - Combine third replica of blocks from a single file to create parity block
  - Remove third replica
  - Apache JIRA HDFS-503
- DiskReduce from CMU
  - Garth Gibson research

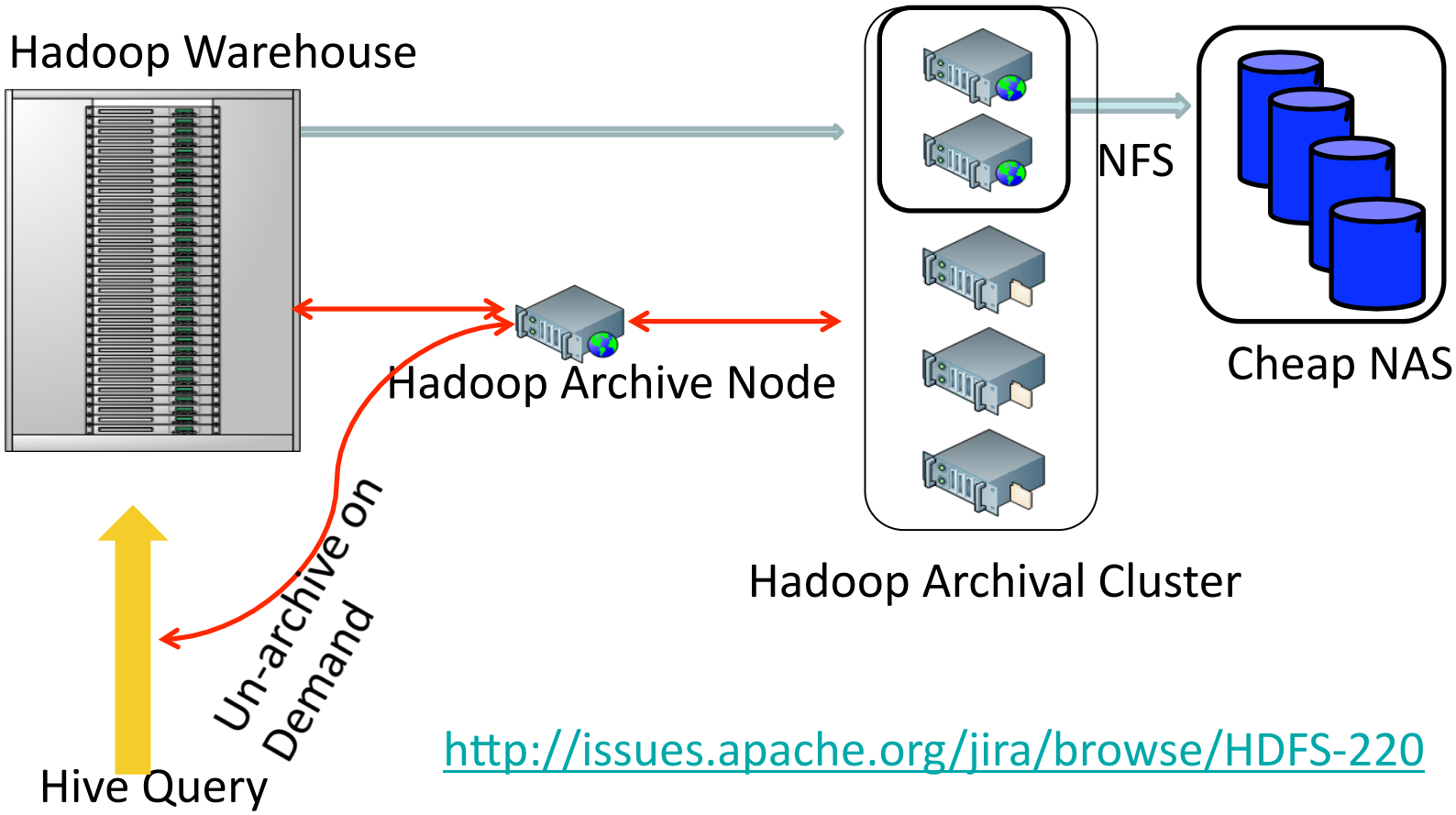


A file with three blocks A, B and C

http://hadoopblog.blogspot.com/2009/08/hdfs-and-erasure-codes-hdfs-raid.html

facebook

Archival: Move old data to cheap storage



http://issues.apache.org/jira/browse/HDFS-220

Dynamic-size MapReduce Clusters

- **Why multiple compute clouds in Facebook?**
  - Users unaware of resources needed by job
  - Absence of flexible Job Isolation techniques
  - Provide adequate SLAs for jobs
- **Dynamically move nodes between clusters**
  - Based on load and configured policies
  - Apache Jira MAPREDUCE-1044

facebook

Resource Aware Scheduling (Fair Share Scheduler)

- **We use the Hadoop Fair Share Scheduler**
  - Scheduler unaware of memory needed by job
- **Memory and CPU aware scheduling**
  - RealTime gathering of CPU and memory usage
  - Scheduler analyzes memory consumption in realtime
  - Scheduler fair-shares memory usage among jobs
  - Slot-less scheduling of tasks (in future)
  - Apache Jira MAPREDUCE-961

facebook

Hive – Data Warehouse

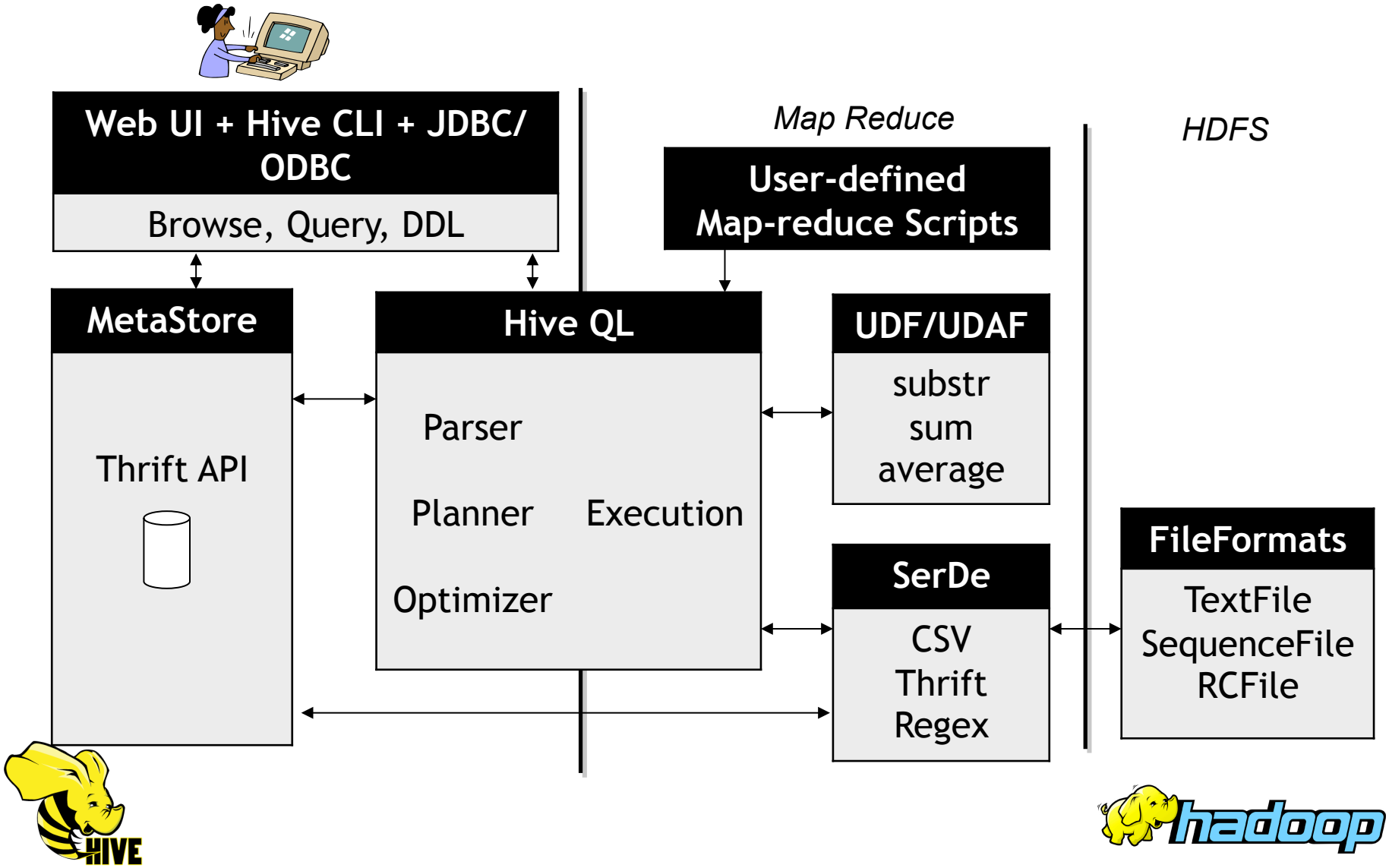
- Efficient SQL to Map-Reduce Compiler

- Mar 2008: Started at Facebook
- May 2009: Release 0.3.0 available
- Now: Preparing for release 0.4.0

- Countable for 95%+ of Hadoop jobs @ Facebook
- Used by ~200 engineers and business analysts at Facebook every month

facebook

Hive Architecture



Hive DDL

- DDL
  - Complex columns
  - Partitions
  - Buckets
- Example
  - ```
    CREATE TABLE sales (
      id INT,
      items ARRAY<STRUCT<id:INT, name:STRING>>,
      extra MAP<STRING, STRING>
    ) PARTITIONED BY (ds STRING)
    CLUSTERED BY (id) INTO 32 BUCKETS;
    ```


    ```

Hive Query Language

- SQL
  - Where
  - Group By
  - Equi-Join
  - Sub query in from clause
- Example
  - ```
    SELECT r.*, s.*
    FROM r JOIN (
      SELECT key, count(1) as count
      FROM s
      GROUP BY key) s
    ON r.key = s.key
    WHERE s.count > 100;
    ```


    ```

Group By

- 4 different plans based on:
  - Does data have skew?
  - partial aggregation
- Map-side hash aggregation
  - In-memory hash table in mapper to do partial aggregations
- 2-map-reduce aggregation
  - For distinct queries with skew and large cardinality

Join

- Normal map-reduce Join
  - Mapper sends all rows with the same key to a single reducer
  - Reducer does the join
- Map-side Join
  - Mapper loads the whole small table and a portion of big table
  - Mapper does the join
  - Much faster than map-reduce join

Sampling

- Efficient sampling
  - Table can be bucketed
  - Each bucket is a file
  - Sampling can choose some buckets
- Example
  - ```
    SELECT product_id, sum(price)
    FROM sales TABLESAMPLE (BUCKET 1 OUT OF 32)
    GROUP BY product_id
    ```


    ```

facebook

Multi-table Group-By/Insert

```
FROM users
INSERT INTO TABLE pv_gender_sum
   SELECT gender, count(DISTINCT userid)
   GROUP BY gender
INSERT INTO
   DIRECTORY '/user/facebook/tmp/pv_age_sum.dir'
   SELECT age, count(DISTINCT userid)
   GROUP BY age
INSERT INTO LOCAL DIRECTORY '/home/me/pv_age_sum.dir'
   SELECT country, gender, count(DISTINCT userid)
   GROUP BY country, gender;
```

File Formats

- TextFile:
  - Easy for other applications to write/read
  - Gzip text files are not splittable
- SequenceFile:
  - Only hadoop can read it
  - Support splittable compression
- RCFile: Block-based columnar storage
  - Use SequenceFile block format
  - Columnar storage inside a block
  - 25% smaller compressed size
  - On-par or better query performance depending on the query

SerDe

- Serialization/Deserialization
- Row Format
  - CSV (LazySimpleSerDe)
  - Thrift (ThriftSerDe)
  - Regex (RegexSerDe)
  - Hive Binary Format (LazyBinarySerDe)
- LazySimpleSerDe and LazyBinarySerDe
  - Deserialize the field when needed
  - Reuse objects across different rows
  - Text and Binary format

UDF/UDAF

- Features:
  - Use either Java or Hadoop Objects (int, Integer, IntWritable)
  - Overloading
  - Variable-length arguments
  - Partial aggregation for UDAF
- Example UDF:

```
public class UDFExampleAdd extends UDF {
  public int evaluate(int a, int b) {
    return a + b;
  }
}
```

facebook

Hive – Performance

Date	SVN Revision	Major Changes	Query A	Query B	Query C
2/22/2009	746906	Before Lazy Deserialization	83 sec	98 sec	183 sec
2/23/2009	747293	Lazy Deserialization	40 sec	66 sec	185 sec
3/6/2009	751166	Map-side Aggregation	22 sec	67 sec	182 sec
4/29/2009	770074	Object Reuse	21 sec	49 sec	130 sec
6/3/2009	781633	Map-side Join *	21 sec	48 sec	132 sec
8/5/2009	801497	Lazy Binary Format *	21 sec	48 sec	132 sec

- QueryA: SELECT count(1) FROM t;
- QueryB: SELECT concat(concat(concat(a,b),c),d) FROM t;
- QueryC: SELECT * FROM t;
- map-side time only (incl. GzipCodec for comp/decompression)
- * These two features need to be tested with other queries.

Hive – Future Works

- Indexes
- Create table as select
- Views / variables
- Explode operator
- In/Exists sub queries
- Leverage sort/bucket information in Join